Searching for Novel Regression Functions

Yuliana Martínez,
Enrique Naredo,
and Leonardo Trujillo
Departamento de Ingeniería Eléctrica y Electrónica
Doctorado en Ciencias de la Computación
Instituto Tecnológico de Tijuana, México
Email: ysaraimr@gmail.com
Email: enriquenaredo@gmail.com
Email:leonardo.trujillo@tectijuana.edu.mx

Edgar Galván-López
Distributed System Group
School of Computer Science and Statistics
Trinity College Dublin, Ireland
Email: edgar.galvan@scss.tcd.ie

***Abstract*—The objective function is the core element in most search algorithms that are used to solve engineering and scientific problems, referred to as the fitness function in evolutionary computation. Some researchers have attempted to bridge this difference by reducing the need for an explicit fitness function. A noteworthy example is the novelty search (NS) algorithm, that substitutes fitness with a measure of uniqueness, or *novelty*, that each individual introduces into the search. NS employs the concept of behavioral space, where each individual is described by a domain-specific descriptor that captures the main features of an individual's performance. However, defining a behavioral descriptor is not trivial, and most works with NS have focused on robotics. This paper is an extension of recent attempts to expand the application domain of NS. In particular, it represents the first attempt to apply NS on symbolic regression with genetic programming (GP). The relationship between the proposed NS algorithm and recent semantics-based GP algorithms is explored. Results are encouraging and consistent with recent findings, where NS achieves below average performance on easy problems, and achieves very good performance on hard problems. In summary, this paper presents the first attempt to apply NS on symbolic regression, a continuation of recent research devoted at extending the domain of competence for behavior-based search.**

***Keywords*—Novelty Search, Behavior-based Search, Genetic Programming, Symbolic Regression**

I. Introduction

The conventional approach towards search and optimization is based on the use of an explicit objective function, this is true for gradient-based methods and heuristic approaches. Moreover, this is also true for evolutionary algorithms (EAs), where the search for better fitness guides the evolutionary process. Fitness is the core element in understanding EA dynamics, with a large amount of research devoted at describing, understanding and analyzing fitness functions [1] and fitness landscapes [2], [3]; such as work done on locality [4], [5] and neutrality [6], [7], [8]. Fitness-based algorithms have achieved impressive results in many domains, for instance just considering the genetic programming (GP) paradigm a long list of examples exist [9]. However, it is noteworthy that the inspiration of EAs comes from a natural process where a set goal or purpose is not present, this process is Darwinian evolution, based on random search operations (mutation and recombination) filtered by non-random natural selection. In other words, Darwinian evolution is an open-ended process while most traditional EAs are not.

However, open-ended techniques have not been completely excluded from EA development; in fact one of the earliest EAs was the open-ended biomorphs system proposed by Richard Dawkins [10]. Moreover, over the years several open-ended systems have been developed, including recent examples in artificial life [11] and interactive search for artistic design [12], [13], [14]. Another prominent example is the novelty search (NS) algorithm proposed by Lehman and Stanley [15], [16], [17], where the explicit objective function is abandoned and selective pressure is instead drawn from a measure of the uniqueness or *novelty* that each individual introduces into the evolving population. Eliminating an explicit objective and promoting and open-ended search could allow for the development of EAs that are based on a more realistic model of natural evolution. Moreover, an open-ended search, particularly one that is based on solution uniqueness such as NS, also has some pragmatic benefits. For instance, fitness based EAs can get trapped or prematurely converge on local optima within a fitness landscape, a problem that in principal should be avoidable by an algorithm such as NS.

The core element of NS is that individuals are not selected based on their fitness, instead the behavior that each individual exhibits is used to describe them, and based on this a measure of novelty is assigned to each. Of course, to do so, a proper behavioral descriptor must be proposed, such that the core elements of an individual's behavior are captured based on the requirements of the application domain, while unnecessary details are abstracted away. This task, however, is not trivial, and care must be taken to propose a proper behavioral descriptor, just like a proper fitness measure is necessary for a traditional fitness-based search. Therefore, until recently, most NS applications have focused on very specific problem domains and applications, such as robotics and interactive evolution. In fact, behavior-based search in general has proven to be quite useful in evolutionary robotics systems, where the concept of behaviors is commonly used and exploited [18], [19]. However, recent works have extended the use of NS to common pattern analysis problems; particularly supervised classification [20] and unsupervised clustering [21] with GP. Both examples present promising results that suggest that behavior-based search can be used in more general domains.

The goal of this paper is to extend the behavior-based search methodology using the NS algorithm to the most common application domain for GP, real-valued symbolic regression problems. Such problems have been extensively studied and analyzed by the GP community [22]. Recently, symbolic regression GP systems have been improved by incorporating the concept of semantics [23], [24], [25], [26], [27], [28], [29], [30], [31]. On the other hand, the present work takes an approach based on behaviors, the difference between both approaches is carefully discussed below. For now, it suffices to say that behaviors are a more general concept than semantics, that can provide a simplified view of what a program is actually doing. Therefore, this work proposes a behavioral descriptor for symbolic regression problems, and results are compared with published results on benchmark tests [32], [24]. The results are consistent with those published in [20], [21], showing that NS performs better when problem difficulty increases, which is what would be expected from a search process that starts with low-quality behaviors and must necessarily explore novel regions in the search space, some of which will contain high-quality solutions; i.e. the search for novelty leads towards quality when problem difficulty is, in some sense, large. These results are indeed promising, another encouraging reason that justifies why behavior-based search strategies should be further explored [20], [21], [33].

The remainder of the paper proceeds as follows. Section II describes previous work, focusing on GP semantics. Then, Section III introduces the concept of behavior-based search. Afterwards, Section IV describes the NS algorithm. Section V presents the proposed symbolic regression behavioral descriptor and discusses how NS is incorporated into a symbolic regression GP system. Experiments and results are presented in Section VI. Finally, a summary of the paper, conclusions and future work are discussed in Section VII.

II. Previous Work

When an EA performs a search, it is well understood that it concurrently operates within three spaces. First, genotypic space or the algorithm's search space, that depends upon the encoding used to represent valid candidate solutions. The genotypic representation is mapped, or decoded, to phenotypic space (also known as problem or decision space), where individual solutions are expressed within the problem domain. Finally, each individual is assigned a performance score in fitness or objective space. Depending on the representations used, genotypic and phenotypic space can be distinct or not [34].

Recently, however, other spaces have started to be analyzed with respect to EAs. Particularly, other descriptions of a solution's performance have been proposed, to either complement or replace an explicit fitness function. Consider the classic GP scenario, where a GP must optimize a given evaluation criterion based on a set of fitness cases. The common approach is for a fitness function to provide a coarse overview of performance, where different performance on different fitness cases is mostly averaged out. Therefore, some researchers have started to develop approaches that can provide a finer grained look of an individual's performance. In the remainder of this section we will summarize the semantics-based approach in GP, that follows this line of reasoning.

A. Semantics in Genetic Programming

Semantics in GP, and its corresponding semantic space, describes the performance of an individual using its raw output vector computed over all the fitness cases of a problem [23], [24], [25], [26], [27], [28], [31]. In other words, given a set of n fitness cases the semantics of a program K is the corresponding set of outputs it produces, normally expressed as a vector $\mathbf{y} \in \mathbb{R}^n$. Semantics is an important concept in GP because (vastly) many genetically (phenotypically) different programs can share the same semantic output. Therefore, it is assumed that the search process must explicitly consider the semantic space of programs to properly conduct the search. Therefore, researchers have used semantics to modify traditional genetic operators [24], [26], [27], focusing on improving the semantic diversity of the evolving population. Moreover, other approaches have been proposed to perform the evolutionary search within semantic space explicitly [25], [28], [31]. In general, all of these works have shown improved results compared to standard GP algorithms, mostly evaluated on symbolic regression problems.

However, strictly focusing on program outputs might not be the best approach for some problem domains. For example, consider the GP classifier based on static range selection (SRS) [35], that functions as follows: for a two class problem and real-valued GP outputs, the SRS classifier is straightforward; if the program output for input pattern $\mathbf{x}$ is greater than zero then the pattern is labeled as belonging to class A, otherwise it is labeled as a class B pattern. In this case, while the semantic space description (as defined above) of two programs might be different (maybe substantially), they can still produce the same high-level classification (consider any two outputs $\mathbf{y}_1, \mathbf{y}_2 \in (0, \infty)$ with $\mathbf{y}_1 \neq \mathbf{y}_2$).

For symbolic regression, a common fitness score (for instance, the average error) for an individual will depend on its semantics. In fact, it can be said that the semantics of a program provides a complementary view of its fitness, each at different levels of abstraction. Normally, fitness is at the coarsest possible level, averaging out all of the performance variations. On the other hand, program semantics provides a fine level of detail, explicitly contemplating every variation in program outputs. In the following section, an intermediate view of program performance is described, based on the concept of behavioral space.

III. Behavior based search

Consider the case of evolutionary robotics (ER) systems. In ER, EAs are normally used to search for robust controllers of autonomous systems [19]. The goal is to find high quality solutions while introducing as little prior knowledge as possible into the objective function, such that the search is performed based on a very high-level definition of the problem [36]. In this scenario, the correspondence between program inputs, outputs and induced actions is less clear. Moreover, in ER fitness evaluation can be performed within real or simulated environments, where noisy sensors and the physical coupling between actuators and the real world can produce a non-injective and non-surjective relation between program output and robot actions.

Therefore, some researchers have begun to consider behavioral space during an evolutionary search [37], [38]. In robotics, the concept of behaviors dates back to the seminal works of Brooks from the 1980's [18]. A behavior is a description β of the way an agent K (program in the GP case) acts in response to a stimulus (or series of) within a particular context $\mathcal{C}$. A context $\mathcal{C}$ includes the description an agent has of its environment and its own internal state, as well as the external environmental conditions in which he is situated. Stated another way, a behavior β is produced by the interactions of agent K, output $\mathbf{y}$ and context $\mathcal{C}$. In behavior-based robotics, for instance, behaviors are described at a very high-level of abstraction by the system designer. In ER, on the other hand, researchers have recently proposed domain-specific numerical descriptors to describe each behavior, allowing them to explicitly consider behavioral space during evolution [38]. The justification for this is evident, given that the objective function is stated as a high-level goal, then population management should also consider the behavioral aspect of evolved solutions. Following this approach, researchers have been able to develop diversity preservation techniques [37], [39] and open-ended search algorithms [15], [17]; for a comprehensive review of behavioral space analysis in ER see [38].

Hence, a behavior defined in this way, can be seen as a higher-level description of the performance of a program, compared to the semantics approach. An individual's behavior is described in more general terms, accounting not only for program output but the context in which the output was produced. While semantics can imply an injective or non-injective relation between input and output, behaviors are more general, allowing for multi-valued functions or many-to-many relations, if only input is considered and context is not. For instance, for the SRS GP classifier described above, context is given by the SRS heuristic rule used to assign class label. This is more evident in the ER examples, where context is given by the environment and internal state of the robot; for instance, depending on whether or not the controller has memory.

In summary, fitness, program semantics, and behavior can be understood as different levels of abstraction of a program's performance, as depicted in Figure 1. At one extreme form of analysis, fitness provides a coarse grained look at performance, a single value (for each criteria) that attempts to capture a global evaluation of what a program does. At the other end of the analysis scale, semantics describe program performance in great detail considering all of the raw program outputs. On the other hand, behavioral descriptors move between fitness and semantics, providing a finer or coarser level of description depending on how behaviors are meaningfully characterized within a particular application domain and the context in which the program operates.

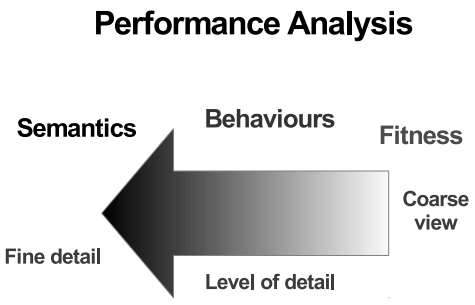


Fig. 1. Conceptual view of how the performance of a program can be analyzed. At one extreme (right hand side) we have fitness-based analysis, a coarse view of performance. Semantics lies at another extreme (left hand side), where a high level of detail is sought. Finally, behavioral analysis is a varying scale that depends on how context is incorporated into the analysis of performance.

IV. Novelty Search

Lehman and Stanley proposed the NS algorithm to eliminate an explicit objective function from an EA [15], [16], [17]. Therefore, the search is not guided by a measure of *quality* for each individual, selective pressure is provided by a measure of *novelty*. The strategy is to measure the amount of novelty each individual introduces into the search with respect to the current population and also individuals found in previous generations.

NS operates based on the concept of solution behaviors, where each individual is described based on the functional behavior it exhibits, as described in the previous section. This description of behavior is captured by a domain dependent descriptor, in such a way that each individual is mapped to a single point in behavioral space. Since, many individuals in the genotypic space express the same behavior and are thus mapped to the same point in behavioral space, the search for novelty is often feasible in practice [15], [16], [17]. Moreover, Lehman and Stanley argue that since the number of simple behaviors for any given problem is relatively small, then the search for novelty must necessarily lead to more complex behavioral patterns.

A known limitation of objective-based search is a tendency to converge and get trapped on local optima, particularly in real-world and multi-modal problems. To overcome this, researchers have incorporated diversity preservation techniques into EAs, such as niching or speciation [40]. However, most proposals are ad-hoc strategies that must attempt to balance exploration and exploitation during the search. Conversely, through the search for novelty alone, diversity preservation introduces the sole selective pressure and can in principle avoid stagnation during the search.

In practice, NS uses a measure of local sparseness around each individual within behavioral space to estimate its novelty, considering the current population and novel solutions from previous generations. An important observation is that such a measure of novelty is dynamic, since it can produce different results for the same individual depending on the population state and search progress at any given generation. The proposed measure of sparseness ρ around each individual K, described by its behavioral descriptor $\boldsymbol{\beta}$, is given by the average distance to the k-nearest neighbors in behavioral space, with k an algorithm parameter, given by

$$\rho(\mathbf{x}) = \frac{1}{k}\sum_{i=0}^{k} dist(\mathbf{x}, \mu_i) \ , \qquad (1)$$

where α_i is the behavioral descriptor of the ith-nearest neighbor of K in behavioral space with respect to distance metric

$dist$, which is a domain-dependent measure of behavioral difference between two descriptors. Given this definition, when the average distance is large, then the individual lies within a sparse region of behavioral space, and it is in a dense region if the measure is small.

An important consideration in NS relates to how neighborhood is defined; the original proposal is to consider the current population and an archive of individuals that were at one time considered to be novel, it is therefore an inter-generational neighborhood. An individual is added to the archive if its sparseness is above a minimal threshold ρ_{min}, the second parameter of the NS algorithm. Finally, at the conclusion of the run after a fixed number of generations, the best solution based on fitness (it is the only moment in which NS uses an explicit fitness function) from both the final population and the archive is returned as the solution found by the search.

Since its proposal in [15], and later works [16], [17], most applications of NS have focused on robotics, such as mobile robot navigation [15], [16], [17], morphology design [41] and gait control [17]. Only until recently has NS been used in general pattern recognition problems, particularly supervised classification [20] and unsupervised clustering [21]. This paper continues to expand the use of the open-ended NS algorithm, applying it to the most common application domain for GP, real-valued symbolic regression.

V. Regression with NS & GP

This paper presents a NS-based GP for real-valued symbolic regression problems, hereafter referred to as NS-GP-R. In regression analysis, a function $f(\mathbf{x})$ is approximated by an operator $K(\mathbf{x})$, considering independent variable $\mathbf{x} \in \mathbb{D}^n \subseteq \mathbb{R}^n$, where $\mathbb{D}^n$ is the domain of $f(\mathbf{x})$. If we consider the training set of fitness cases as X, then the goal is to find an operator K that minimizes $\|f(\mathbf{x}) - K(\mathbf{x})\|$. However, to use NS the performance of $K(\mathbf{x})$ must not be expressed as an error over all fitness cases, but as a description of how K behaves over the entire set of fitness cases.

A. Behavioral Descriptor

Since the goal of a symbolic regression problem is straightforward, to minimize the error between the desired output and the proposed approximation, then a behavioral descriptor for symbolic regression must consider the concept of error in some way. The descriptor must provide a sufficiently detailed description of how an individual behaves over all fitness cases. Moreover, the descriptor should abstract away unnecessary details, and focus on describing how unique a particular individual is relative to others within the population.

ϵ-descriptor: For a regression problem, let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be the function to be approximated, $X = \{(\mathbf{x}_1, f(\mathbf{x_1})), (\mathbf{x}_2, f(\mathbf{x_2})), ..., (\mathbf{x}_L, f(\mathbf{x_L}))\}$ the set of fitness cases (input-output tuples), K_i a valid GP individual, and $\mathbf{e}_i = \{e_{i,1}, e_{i,2}, ..., e_{i,L}\}$ be the error vector of K_i with $e_{i,j} = |f(\mathbf{x_j}) - K_i(\mathbf{x_j})|$; then the ϵ-descriptor $\boldsymbol{\beta}^{\epsilon}_i$ of K_i is a binary vector of size L that is constructed as follows.

For simplicity and without loss of generality, consider $L = 1$. Then, at a generation t with population P, sort P in ascending order based on e_i and compute the order statistic p of the set of error vectors $E = \{\mathbf{e}_i | \forall K_i \in P\}$, where $p = \frac{|P|}{h}$ with h an algorithm parameter. Then, set a bounding value $\epsilon_t = min(\epsilon_{t-1}, e_p)$ if $t > 0$ and $\epsilon_t = e_p$ otherwise, such that $\beta^{\epsilon}_i = 1$ for all K_i with $i \leq p$ and $\beta^{\epsilon}_i = 0$ otherwise.

Algorithm 1 Behavior descriptor β^{ϵ}

Require: Input
// $x \in \mathbb{R}^n$
// $f(x)$: symbolic function
// K : GP function
Ensure: Output binary vector $\boldsymbol{\beta}^{\epsilon}$
// gen: generation number
for $t = 0 : gen$ **do**
  // pop: population number
  for $i = 1 : pop$ **do**
    // L: sample size
    for $j = 1 : L$ **do**
      // $E_{i,j} = \{e_{i,j}\}$: error matrix
      $e_{i,j} \Leftarrow |f(x_j) - K_i(x_j)|$
    end for
  end for
  // descending sort by column, $E = e_{i,j}$
  $E \Leftarrow$ **sort** $(E,\ j,\ descend)$
  // order statistic p (read text)
  $p \Leftarrow \frac{\lfloor P \rfloor}{h}$
  // vector row at threshold h
  $\boldsymbol{\epsilon}_{t+1} \Leftarrow \mathbf{e}_p$
  if $\boldsymbol{\epsilon}_{t+1} > \boldsymbol{\epsilon}_t \ \&\ t \neq 0$ **then**
    // epsilon vector update
    $\boldsymbol{\epsilon}_{t+1} \Leftarrow \boldsymbol{\epsilon}_t$
  end if
  // individual descriptor loop
  for $i = 1 : pop$ **do**
    for $j = 1 : L$ **do**
      if $e_{ij} \leq \epsilon_{t+1,j}$ **then**
        $\beta^{\epsilon}_{i,j} \Leftarrow 1$
      else
        $\beta^{\epsilon}_{i,j} \Leftarrow 0$
      end if
    end for
  end for
end for

For example consider $h = 10$, then the ϵ-descriptor $\boldsymbol{\beta}^{\epsilon}_i$ of K_i identifies the fitness cases j on which individual K_i is in the best 10-percentile of the population, when $\beta^{\epsilon}_{i,j} = 1$. Conversely, if $\beta^{\epsilon}_{i,j} = 0$, this means that the individual performs in the worst 90% of the population. A graphical description of this process is shown in Figure 2. Moreover, a description of the algorithm used to compute the ϵ-descriptor is given in Algorithm 1.

B. NS Modifications

The behavioral descriptor presented above is constructed based on the performance of each individual relative to the rest of the population. Therefore, the descriptor assigned to any given individual will depend on the population to which it belongs. In other words, the context in which a descriptor is computed varies as a function of the current population. This represents a slight modification to the original NS implementa-

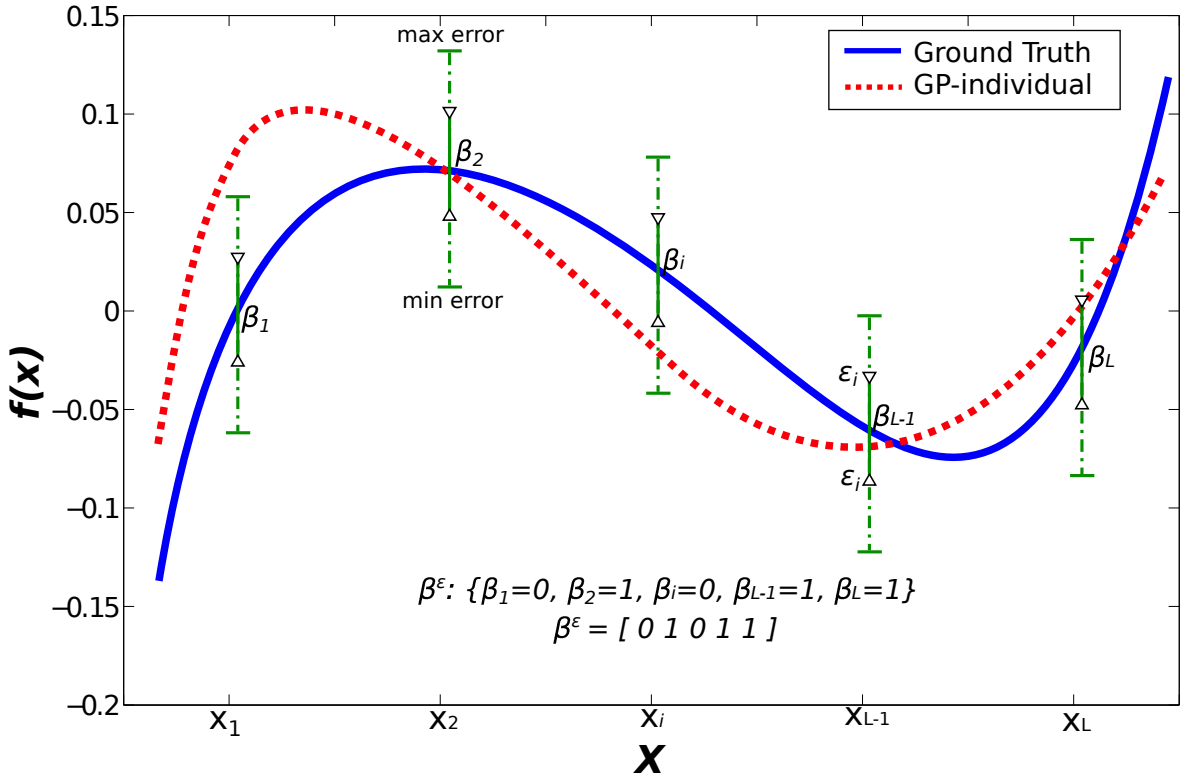


Fig. 2. Graphical depiction of how the ϵ-descriptor descriptor β^{ϵ} is constructed.

tion. However, it is consistent with the idea that the uniqueness of an individual depends on the search progress at the moment in which the individual was found.

This design choice does raise a problem when sparseness is computed, since the descriptors of individuals stored in the archive will, with high probability, not represent a consistent behavioral descriptor with respect to the individuals in the current population. Therefore, the second modification made to the NS algorithm is to omit the archive when sparseness is computed. The lack of an archive can lead the canonical NS to perform backtracking during the search, cycling within behavioral space without any real progress. However, this problem is avoided by not allowing ϵ to increase in magnitude. Since, for most problems, the initial random population will mostly contain individuals with a large error vector, in the initial generations the ϵ bounds will tend to be high. Then, with selective preference given to individuals with unique behaviors and given the way in which the ϵ-descriptor is constructed, the algorithm will be biased towards individuals with descriptors that contain a large proportion of ones since most low performance solutions will have behavioral descriptors with a large proportion of zeros; i.e., it will be biased towards individuals that achieve a low error on many fitness cases. Therefore, since ϵ cannot increase over generations, then selective pressure will push the search towards novel individuals that exhibit a low error on many fitness cases. Nonetheless, the archive is still used to save promising individuals across generations, and is used to select the best solution at the end of the run.

VI. Experiments

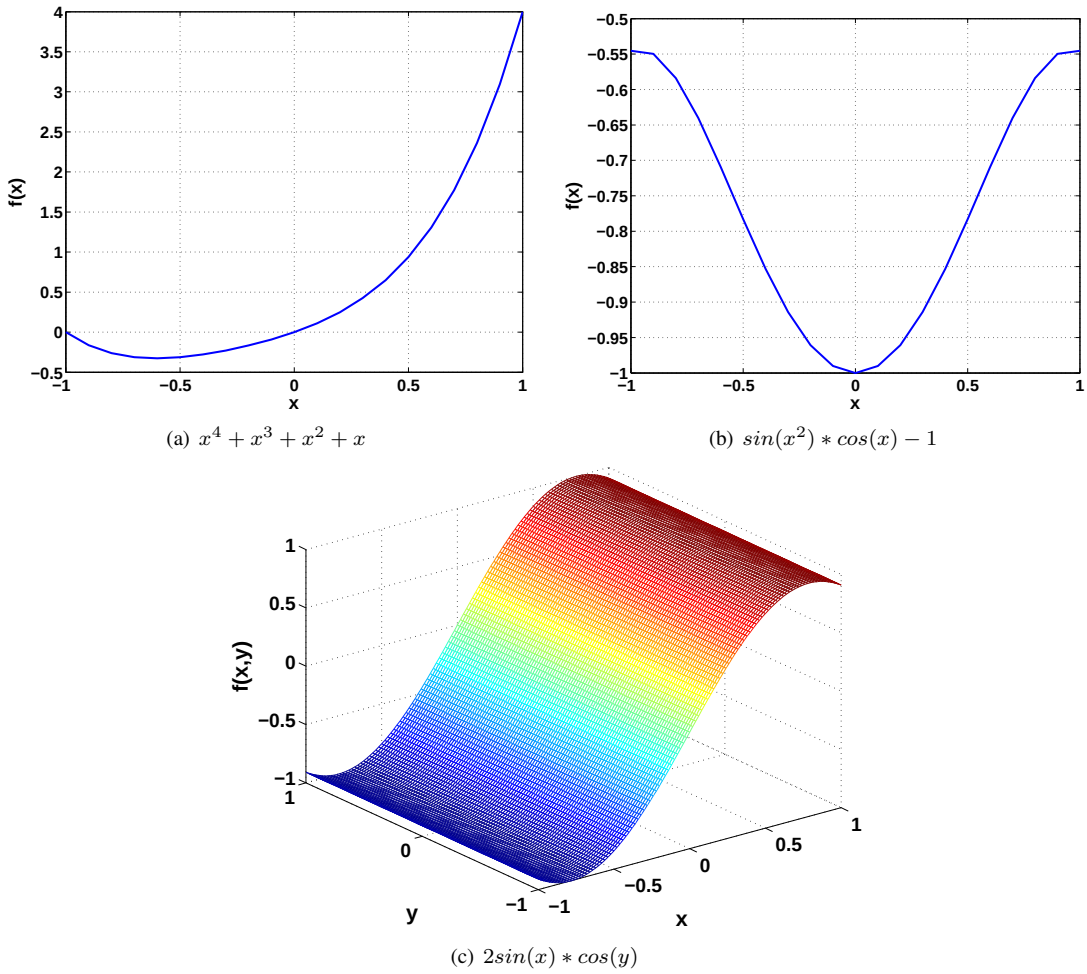
The proposed NS-GP-R algorithm is evaluated on three benchmark problems, suggested by [32] and proposed in [24]. Moreover, for comparative purposes, the algorithm is compared to the results published in [24], that use a standard GP, hereafter referred to as SGP, and a GP with the Semantic Similarity-based Crossover (SSC) also proposed in [24].

A. Test Problems and Parametrization

The three symbolic regression problems are given in Table I. The problems were chosen based on the difficulty the problems posed to the methods published in [24], and they are ordered from the easiest to the most difficult in descending order [1]. The two easier problems have one independent variable, while the harder problem has two independent variables; Figure 3 shows the ground truth function in each problem. Table I specifies the desired function and the manner in which the training set (fitness cases) and testing set are constructed, using the same random procedure in both.

The GP setup and parameters are given in Table II, using a similar configuration to [24]. The NS-GP-R algorithm was coded using GPLAB, a GP Matlab toolbox [42].

[1]It is not claimed that this ordering implies any deeper understanding of the intrinsic difficulty of the problem, it is only based on the performance of the algorithms compared in [24].



(a) $x^4 + x^3 + x^2 + x$

(b) $sin(x^2) * cos(x) - 1$

(c) $2sin(x) * cos(y)$

Fig. 3. Benchmark symbolic regression problems.

TABLE I. BENCHMARK PROBLEMS FROM [24].

Problem	Function	Fitness/Test cases
f_1	$x^4 + x^3 + x^2 + x$	20 random points $\subseteq [-1, 1]$
f_2	$sin(x^2) * cos(x) - 1$	20 random points $\subseteq [-1, 1]$
f_3	$2sin(x) * cos(y)$	100 random points $\subseteq [-1, 1]x[-1, 1]$

TABLE II. PARAMETERS FOR THE GP-BASED SEARCH.

Parameter	Description
Population size	200 individuals.
Generations	100 generations.
Initialization	*Full*,with 6 levels.
Crossover	One-point crossover.
Mutation	Subtree mutation.
Operator probabilities	Crossover $p_c = 0.9$, Mutation $p_\mu = 0.05$.
Function set	$(+, -, \times, \div, sin, cos, exp, log)$.
Terminal set	$x, 1$ for single variable problems and x, y for bivariable problem.
Hard maximum depth	15 levels.
Selection	Tournament of size 6.
NS nearest neighbors	$k = 15$.
Sparseness threshold	$\rho = 3$ for single variable problems and $\rho = 13$ for bivariable problem.
ϵ-descriptor threshold	$h = 10$.
Number of runs per problem	30.

B. Experimental Results

After executing the algorithm 30 times on each problem, the following results are obtained. First, Figure 4 presents a boxplot analysis of the performance of NS-GP-R on each benchmark problem, showing the best error on the test-set achieved in each run. For comparative purposes Table III presents a comparison of the average error of NS-GP-R on each benchmark, along with the average error reported in [24] for SGP and SSC. The table shows an interesting trend, NS performs worse on the easier problem, performs about equally on the intermediate problem, and outperforms all methods on the most difficult problem [2]. These results are interesting but not unexpected, they are consistent with recent results achieved by NS on classification and data clustering [20], [21]. The trend is apparent, NS works best when the problems are difficult, because in this scenario, novelty indeed leads towards quality, given that the initial random populations mostly contain individuals that perform poorly. Therefore, in difficult problems the selection pressure in novelty search can lead towards individuals that achieve good performance. In other words, when the gradient for novelty is positively

[2]Results are compared based only on average error, no statistical tests are performed because only the published results are being used for comparison.

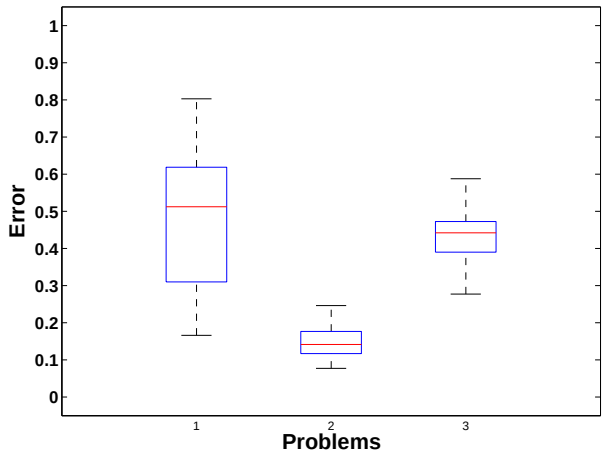


Fig. 4. Boxplot analysis of NS-GP-R performance on each benchmark problem, showing the best test-set error achieved in each run.

TABLE III. COMPARISON OF NS-GP-R WITH TWO CONTROL METHODS REPORTED IN [24], SSC AND SGP; VALUES ARE THE MEAN ERROR COMPUTED OVER ALL RUNS.

Problem	NS-GP-R	SSC	SGP
1	0.47	0.15	0.26
2	0.14	0.10	0.21
3	0.42	1.53	2.26

correlated with the gradient for fitness, then NS can indeed find high performance solutions.

VII. CONCLUSIONS

This work presents a NS-based GP algorithm for the classical GP problem of symbolic regression. The NS algorithm is an open-ended EA, where an explicit fitness function is not used to guide the search, instead a measure of the uniqueness of each individual provides the selective pressure during evolution. In this respect, the paper represents the first attempt to apply NS in this domain, one of the most common engineering and scientific problems. To achieve this, however, the concept of behavioral space needed to be adapted for it to be applied in this domain. Therefore, a behavioral descriptor was proposed called ϵ-descriptor, that emphasizes the main characteristics of program behavior while abstracting away unnecessary details, in order to properly describe the performance that each GP individual exhibits with respect to the rest of the population.

Results are encouraging and consistent with recent proposals that expand the use of the NS algorithm to other mainstream areas. The proposed NS-based GP was compared with recently published results on three benchmark problems that are currently suggested for GP evaluation within the community [32]. NS shows a consistent trend, it achieved quite bad performance on easy problems, and performs substantially better on difficult ones, results that are similar to those published in [20], [21]. For real-world scenarios these results are promising, since most interesting problems are difficult, not easy. The reason for these results can be inferred from the nature of the NS algorithm. Random solutions in the initial population mostly exhibit bad performance, thus the selective pressure towards good solutions increases during the search for novelty. On the other hand, for easier problems the exploration performed by NS is mostly counterproductive, since random solutions might provide good initial approximations and all that is lacking is an exploitative search. Therefore, when random solutions have a high fitness then novelty could easily lead the search towards worse results. Conversely, when the problem is difficult and random solutions are of low fitness, then the search for novelty will lead towards high quality.

Future work will center on exploring further experimental tests in this domain, comprehensively evaluating different parameterizations and performance achieved on other benchmarks and real-world problems, comparing the algorithm with other GP systems for symbolic regression. Moreover, it is imperative to provide a deep comparison between the proposed behavior-based search strategy and recent semantics-based approaches, a comparison that goes beyond merely experimental results, but a detailed analysis of the main algorithmic differences between both approaches and their effects on search. Finally, future work should also study how NS affects the bloat phenomenon during a GP search. The reason for this is actually very intuitive, given that the main cause for bloat is the bias introduced by the search for better fitness, what is known as the fitness-causes-theory [43], [44]. Recent studies of NS applied to pattern recognition problems [20], [21] has produced results that suggest that the elimination of an explicit fitness function can curtail bloating during a GP run [33]. However, since the algorithm and behavioral descriptors used in [20], [21], [33] differ on some key aspects with respect to the proposal described in this paper, then further analysis and experimental validations is required.

Acknowledgments.: Funding for this work provided CONACYT (Mexico) Basic Science Research Project No. 178323 and DGEST (Mexico) Research Project No. TIJING-2012-110. First and second author are supported by CONACYT (Mexico) scholarships, respectively scholarships No. 226981 and No. 232288. The fourth author wishes to acknowledge Science Foundation Ireland (SFI) under the Principal Investigator research program 10/IN.1/I2980 "Self-organizing Architectures for Autonomic Management of Smart Cities" and by SFI grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie) for their support to conduct this research.

REFERENCES

[1] L. Vanneschi, M. Castelli, and L. Manzoni, "The k landscapes: a tunably difficult benchmark for genetic programming," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, ser. GECCO '11. New York, NY, USA: ACM, 2011, pp. 1467–1474.

[2] P. F. Stadler and C. R. Stephens, "Landscapes and effective fitness," *Comments on Theoretical Biology*, vol. 8, no. 4, pp. 389–431, 2003.

[3] W. Hordijk, "A measure of landscapes," *Evol. Comput.*, vol. 4, no. 4, pp. 335–360, December 1996.

[4] E. Galván-López, J. McDermott, M. O'Neill, and A. Brabazon, "Defining locality in genetic programming to predict performance," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, 2010, pp. 1–8.

[5] ——, "Towards an understanding of locality in genetic programming," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, ser. GECCO '10. New York, NY, USA: ACM, 2010, pp. 901–908.

[6] E. Galván-López, S. Dignum, and R. Poli, "The effects of constant neutrality on performance and problem hardness in gp," in *Proceedings of the 11th European conference on Genetic programming*, ser. EuroGP'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 312–324.

[7] E. Galván-López and R. Poli, "An empirical investigation of how and why neutrality affects evolutionary search," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, ser. GECCO '06. New York, NY, USA: ACM, 2006, pp. 1149–1156.

[8] R. Poli and E. Galván-López, "The effects of constant and bit-wise neutrality on problem hardness, fitness distance correlation and phenotypic mutation rates," *IEEE Trans. Evolutionary Computation*, vol. 16, no. 2, pp. 279–300, 2012.

[9] J. Koza, "Human-competitive results produced by genetic programming," *Genetic Programming and Evolvable Machines*, vol. 11, no. 3, pp. 251–284, 2010.

[10] R. Dawkins, *Climbing Mount Improbable*. W.W. Norton & Company, 1996.

[11] C. Ofria and C. O. Wilke, "Avida: a software platform for research in computational evolutionary biology," *Artif. Life*, vol. 10, no. 2, pp. 191–229, 2004.

[12] T. Kowaliw, A. Dorin, and J. McCormack, "Promoting creative design in interactive evolutionary computation," *Evolutionary Computation, IEEE Transactions on*, vol. 16, no. 4, pp. 523 –536, 2012.

[13] M. Garcia, L. Trujillo, F. Fernández-de Vega, J. Merelo, and G. Olague, "Evospace: A distributed evolutionary platform based on the tuple spacemodel," in *Proceedings from EvoApplications '12*, ser. LNCS, A. Esparcia-Alcázar, Ed. Springer-Verlag, 2013, pp. 499–508.

[14] L. Trujillo, M. Garciía-Valdez, F. Férnandez-de Vega, and J. Merelo, "Fireworks: Evolutionary art project based on evospace-interactive," in *to appear Proceedings of the 15th Congress on Evolutionary Computation*, ser. CEC'13. IEEE Press, 2013.

[15] J. Lehman and K. O. Stanley, "Exploiting open-endedness to solve problems through the search for novelty," in *Proceedings of the Eleventh International Conference on Artificial Life, Cambridge, MA*, ser. ALIFE XI. MIT Press, 2008.

[16] ——, "Efficiently evolving programs through the search for novelty," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, ser. GECCO '10. ACM, 2010, pp. 837–844.

[17] ——, "Abandoning objectives: Evolution through the search for novelty alone," *Evol. Comput.*, vol. 19, no. 2, pp. 189–223, 2011.

[18] R. A. Brooks, *Cambrian intelligence: the early history of the new AI*. Cambridge, MA, USA: MIT Press, 1999.

[19] S. Nolfi and D. Floreano, *Evolutionary Robotics: The Biology,Intelligence,and Technology*. Cambridge, MA, USA: MIT Press, 2000.

[20] E. Naredo, L. Trujillo, and Y. Martínez, "Searching for novel classifiers," in *Proceedings from the 16th European Conference on Genetic Programming, EuroGP 2013*, ser. LNCS, vol. 7831. Springer-Verlag, 2013, pp. 145–156.

[21] E. Naredo and Trujillo, "Searching for novel clustering programs," in *to appear in Proceeding from the Genetic and Evolutionary Computation Conference, GECCO 2013*. ACM, 2013.

[22] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd, 2008.

[23] N. F. McPhee, B. Ohs, and T. Hutchison, "Semantic building blocks in genetic programming," in *Proceedings of the 11th European conference on Genetic programming*, ser. EuroGP'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 134–145.

[24] N. Q. Uy, N. X. Hoai, M. O'Neill, R. I. Mckay, and E. Galván-López, "Semantically-based crossover in genetic programming: application to real-valued symbolic regression," *Genetic Programming and Evolvable Machines*, vol. 12, no. 2, pp. 91–119, 2011.

[25] A. Moraglio, K. Krawiec, and C. G. Johnson, "Geometric semantic genetic programming," in *Proceedings of the 12th international conference on Parallel Problem Solving from Nature - Volume Part I*, ser. PPSN'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 21–31.

[26] L. Beadle and C. Johnson, "Semantically driven crossover in genetic programming," in *Proceedings of the Tenth Conference on Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, ser. CEC'08. IEEE Press, 2008, pp. 111–116.

[27] L. Beadle and C. G. John'son, "Semantically driven mutation in genetic programming," in *Proceedings of the Eleventh Conference on Congress on Evolutionary Computation*, ser. CEC'09. IEEE Press, 2009, pp. 1336–1342.

[28] K. Krawiec and T. Pawlak, "Locally geometric semantic crossover: a study on the roles of semantics and homology in recombination operators," *Genetic Programming and Evolvable Machines*, vol. 14, no. 1, pp. 31–63, 2013.

[29] E. Galván-López, B. Cody-Kenny, L. Trujillo, and A. Kattan, "Using semantics in the selection mechanism in genetic programming: a simple method for promoting semantic diversity," in *IEEE Congress on Evolutionary Computation*. IEEE, 2013.

[30] N. Uy, M. O?Neill, N. Hoai, B. Mckay, and E. Galv
'an-López, "Semantic similarity based crossover in gp: The case for real-valued function regression," in *Artifical Evolution*, ser. Lecture Notes in Computer Science, P. Collet, N. Monmarch, P. Legrand, M. Schoenauer, and E. Lutton, Eds. Springer Berlin Heidelberg, 2010, vol. 5975, pp. 170–181.

[31] L. Vanneschi, M. Castelli, L. Manzoni, and S. Silva, "A new implementation of geometric semantic gp and its application to problems in pharmacokinetics," in *Proceedings of the 16th European conference on Genetic Programming*, ser. EuroGP'13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 205–216.

[32] J. McDermott, D. R. White, S. Luke, L. Manzoni, M. Castelli, L. Vanneschi, W. Jaskowski, K. Krawiec, R. Harper, K. De Jong, and U.-M. O'Reilly, "Genetic programming needs better benchmarks," in *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, ser. GECCO '12. New York, NY, USA: ACM, 2012, pp. 791–798.

[33] L. Trujillo, E. Naredo, and Y. Martínez, "Preliminary study of bloat in genetic programming with behavior-based search," in *to appear EVOLVE - A bridge between Probability, Set Oriented Numerics, and Evolutionary Computation III*, ser. EVOLVE'13. Springer-Verlag, 2013.

[34] E. Galván-López, J. Mcdermott, M. O'Neill, and A. Brabazon, "Defining locality as a problem difficulty measure in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 12, no. 4, pp. 365–401, 2011.

[35] M. Zhang and W. Smart, "Using gaussian distribution to construct fitness functions in genetic programming for multiclass object classification," *Pattern Recogn. Lett.*, vol. 27, no. 11, pp. 1266–1274, 2006.

[36] A. L. Nelson, G. J. Barlow, and L. Doitsidis, "Fitness functions in evolutionary robotics: A survey and analysis," *Robot. Auton. Syst.*, vol. 57, no. 4, pp. 345–370, 2009.

[37] L. Trujillo, G. Olague, E. Lutton, and F. F. De Vega, "Discovering several robot behaviors through speciation," in *Proceedings of the 2008 conference on Applications of evolutionary computing*, ser. Evo'08. Springer-Verlag, 2008, pp. 164–174.

[38] J. B. Mouret and S. Doncieux, "Encouraging behavioral diversity in evolutionary robotics: An empirical study," *Evol. Comput.*, vol. 20, no. 1, pp. 91–133, 2012.

[39] L. Trujillo, G. Olague, E. Lutton, F. Fernndez de Vega, L. Dozal, and E. Clemente, "Speciation in behavioral space for evolutionary robotics," *Journal of Intelligent & Robotic Systems*, vol. 64, no. 3-4, pp. 323–351, 2011.

[40] S. W. Mahfoud, "Niching methods for genetic algorithms," Ph.D. dissertation, Champaign, IL, USA, 1995.

[41] J. Lehman and K. O. Stanley, "Evolving a diversity of virtual creatures through novelty search and local competition," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, ser. GECCO '11. ACM, 2011, pp. 211–218.

[42] S. Silva and J. Almeida, "Gplab–a genetic programming toolbox for matlab," in *Proceedings of the Nordic MATLAB conference*, L. Gregersen, Ed., 2003, pp. 273–278.

[43] W. B. Langdon and R. Poli, "Fitness causes bloat," in *Proceedings of the Second On-line World Conference on Soft Computing in Engineering Design and Manufacturing*. Springer-Verlag, 1997, pp. 13–22.

[44] ——, "Fitness causes bloat: Mutation," in *Proceedings of the First European Workshop on Genetic Programming*, ser. EuroGP '98. London, UK, UK: Springer-Verlag, 1998, pp. 37–48.